New Trends in Software Methodologies, Tools and Techniques
H. Fujita et al. (Eds.)
IOS Press, 2014

doi:10.3233/978-1-61499-434-3-256

Integration of Safety Risk Assessment Techniques into Requirement Elicitation

Eileen YEOW[1] and Yin KIA CHIAM
Faculty of Computer Science and Information Technology, University of Malaya, 50603 Kuala Lumpur, Malaysia

Abstract. Incomplete and incorrect requirements may cause the safety-related software systems to fail to achieve their safety goals. It is crucial to ensure software safety by identifying proper software safety requirements during the requirements elicitation activity. Practitioners apply various Safety Risk Assessment Techniques (SRATs) to identify, analyze and assess safety risk. Nevertheless, there is a lack of guidance on how appropriate SRATs and safety process can be integrated into requirements elicitation activity to bridge the gap between the safety and requirements engineering practices. In this research, we proposed an Integration Framework that integrates safety activities and techniques into existing requirements elicitation activity.

Keywords. Safety risk assessment techniques; elicitation; software safety process; requirements engineering; process integration

Introduction

Over recent years, there has been a drastic increase in the number of software usage in safety-related systems. Several safety-related systems such as traffic control, medical devices, smart vehicles, nuclear power plant or railway signaling systems depend on software to manage the safety-related functionalities [1-4]. The demand for software in safety-related system is expected to increase in the following years which turned out to be a challenge in the software safety and requirements engineering processes [5]. Potential software safety hazards may lead to environmental problems, serious injury to people or a loss of human life. It can be seen from several examples such as the radiation overdose in THERAC-25 [4], the explosion of Ariane 5 shortly after the launch [3], and thousands of software errors were found in Voyager and Galileo spacecraft [6]. These incidents are caused by software requirements issues such as having an improper requirements specification, miscommunication of the requirements between operators and the development teams [1,3,4,6]. As a result, software safety becomes a vital concern throughout the requirements engineering process.

A recent survey [7] conducted by the industrial safety experts shows that there is a strong interrelation between safety and requirements engineering (RE) whereby the requirements defined in RE process are handed to the safety engineers for safety hazard analyses. These two processes collaborate iteratively once there are changes in the requirements. Thus, there is a need for a better collaboration between the safety

[1] Corresponding Author: Eileen Yeow, Faculty of Computer Science & Information Technology, E-University of Malaya, 50603 Kuala Lumpur; E-mail: eileeny@um.edu.my.

engineering teams and the requirements engineering teams to transform the safety goals into appropriate safety-related requirements and also to verify and validate requirements specifications from safety perspectives.

In brief, the overall RE [8-10] process consists of five main activities: requirements elicitation, requirements negotiation and requirements documentation, requirements verification and validation (V&V) and requirements management. These RE activities worked intertwiningly to produce system requirements. Software safety risks assessment and software safety requirements elicitation are two key elements to ensure the overall safety of software intensive systems. Safety Risk Assessment Techniques (SRATs) are used to identify, analyse system hazards and evaluate safety risks during requirements elicitation. SRATs are applied to ensure that potential safety hazards that pose risks to the software intensive system are identified, mitigated and eliminated. The safety engineers select and apply various SRATs based on the guidelines from the existing standards or specific requests from their clients. For example, SRATs such as Fault Tree Analysis (FTA), Failure Modes and Effects Analysis (FMEA), Hazard and Operability Study (HAZOP) are well-known techniques for assessing software hazards and risk of the safety-related systems. The work products developed from the safety risk analysis performed by the safety engineering team are used as the inputs by the requirements engineering team to elicit safety-related requirements. Although there are a variety of SRATs available, not all of the techniques are suitable to be applied during requirements elicitation. Therefore, by integrating appropriate SRATs into requirements elicitation activity, this will help both teams working together to develop safety-related requirements.

This research aims to help safety and requirements engineers to have a better understanding of the relationships between SRATs, software safety process and requirements elicitation activity. A framework is proposed to integrate the software safety activities and SRATs into the existing elicitation activity. The proposed work, Safety Risk Assessment Techniques in Requirements Elicitation (SaTRE) Integration Framework is derived by reviewing literature and conducting a survey to identify and select appropriate SRATs and activities that are relevant to software safety requirements elicitation. Subsequently, the selected SRATs, safety activities and steps were mapped into three stages of the requirements elicitation activity (i.e. pre-elicitation, midst of elicitation, and post-elicitation). The framework is evaluated by experts and researchers from requirements and safety engineering domains.

The remainder of this paper is organised as follows. Section 1 discusses work related to this research. Section 2 discusses the selection of SRATs and safety process activities. Section 3 presents the proposed framework. Section 4 discusses limitations and applicability of the framework arising from the evaluation. Finally, Section 5 presents conclusions and discusses future research.

1. Related work

A recent work by Firesmith [11] describes the importance of integrating defensibility (i.e. safety and security) quality into requirements engineering and also highlights the needs of having proper concepts, techniques, tools and expertise to support the collaboration between safety, security, and requirements engineers. Kotonya and Sommerville [12] propose an integration of safety analysis and VORD (Viewpoint-Oriented Requirements Definition) which illustrates how the VORD process model can

improve the process of deriving safety requirements. Besides emphasizing that safety should be built into the early software development process, the authors also show the concept of collaboration between safety and RE. However, currently both works have not considered integrating the SRATs into requirements elicitation activity.

There are some existing safety standards such as IEC61508 [13] and IEC62279 [14] that provides guidelines for SRATs and also the inputs (information or artefacts) needed by and the outputs (the temporary, intermediate or final products) created or modified in each software development phase. Although SRATs have been recommended in these standards, there is still a lack of guidance to support safety and requirements engineers to understand how these techniques can be applied during the requirements phase to elicit software safety requirements. Software safety handbook [15] and software safety guidebook [16] provide the overall software safety analysis activities with inputs and outputs for each SRATs but do not provide further elaboration between safety analysis activities and techniques and also the requirements elicitation activity.

Allenby and Kelly combine the concept of use case scenarios from Unified Modelling Language (UML) together with the Functional Hazard Assessment (FHA) technique to derive the requirements for safety critical aeroengine control systems [17]. This research focuses on utilising the output produced by the elicitation activity (i.e. the use case scenarios) with the FHA approach. More detailed modelling of the process elements (e.g. inputs, outputs, roles, activities) can help both requirements and safety engineering teams to have a better understanding of the relationship between the SRATs and the software safety-related requirements elicitation activity.

2. Selection of SRATs and Activities

The research methodology that we used in selecting SRATs consists of the following three phases.

2.1. Phase I: Identifying and Selecting SRATs and Activities

In Phase I, we have reviewed the safety standards and literature [14,17,31-33] to identify a list of SRATs and safety activities with regard to software requirements elicitation. Next, we have chosen SRATs and activities that are suitable for the integration purpose. There are many SRATs suggested by safety standards and also literature to assure software safety. The criteria that we used to determine the appropriateness of the SRATs for software safety requirements elicitation are as follows:-

- Hazard Identification (HAZID) - Techniques that can be used to identify potential hazards in the software safety systems
- Hazard Analysis (HA) - Techniques that focus more on assessing the potential effects and causes of hazardous events, the likelihood of each hazard's occurrence, and analyses the mitigation steps and preventative approaches.

Based on the aforementioned criteria, Table 1 shows the SRATs that are selected for requirements elicitation activity. These techniques must exist in more than one paper to prove that the techniques are widely used in practice.

Table 1. List of SRATs

Safety Risk Assessment Technique (SRAT)	HAZID	HA	Basic Reference(s)
Accident Model	√	√	[25, 26]
Cause Consequence Analysis (CCA)		√	[14, 24]
Common Cause Failure Analysis (CCF)	√	√	[14, 24]
Data Flow Diagram (DFD)	√	√	[14, 24]
Decision Tables (Truth Tables)		√	[24, 40]
Event Tree Analysis (ETA)		√	[14, 24]
Failure Modes and Effects Analysis (FMEA) / Failure Modes, Effects, Criticality Analysis (FMECA)	√	√	[14, 24]
Fault Tree Analysis (FTA)		√	[14, 24]
Finite State Machine/State Transition Diagrams (FSM/STD)		√	[14, 24]
Formal Methods (e.g. Time Petri Nets)		√	[14, 24]
Hazard and Operability Study (HAZOP)	√	√	[14, 24]
Hazard indices	√	√	[29, 30]
Inspection	√	√	[14, 24]
Markov Model		√	[14, 24]
Misuse Case	√	√	[27, 28]
Preliminary Hazard Analysis (PHA)		√	[4, 18, 20]
Prototyping	√	√	[14, 24]
Safety Checklist	√	√	[23, 24]
What-if	√	√	[21, 22]
Walkthrough	√	√	[14, 24]

Software safety process can be customised to meet specific organisation or project needs. However, in general there are several common activities that must be performed to ensure that the requirements conform to the safety standards [31]. Similar to the RE process, the software safety process also has its own systematic way of capturing safety-related requirements. The main activities of the software safety process that should be part of the elicitation activity are identified and described as follows:

2.1.1. Software Safety Planning

The process starts with a software safety planning which identifies the safety goals [11]. For instance, a company plans to reduce accident rate by 15 percent in the following five years. This activity involves the planning of safety tasks performed in the project and hence produce a safety plan. The objectives of the safety plan are defined according to the safety goals.

2.1.2. Capturing top-level safety-related requirements

Top-level requirements include the overall system requirements such as hardware, software, external devices and constraints of the system. To capture the top–level safety-related requirements, the stakeholders need to perform the following steps:-

- Identify safety-related computer system function and its description such as safety functional requirements, safety significant requirements, and constraints.
- Perform hazard analysis from the scenarios and functional requirements [19].
- Perform risk assessment after the potential hazards have been identified. Each hazard will be analysed to determine the likelihood of occurrences and the level of risks.

- Categorise requirements to specific level, for example the requirements can be classified into system, hardware, or software levels to show the overall association in the requirements.
- Propose mitigation approaches to each requirement respective to risk level.

2.1.3. Capturing design-level safety-related requirements

Capturing design-level safety-related requirements include the following steps:-

- Perform software specific hazard analysis on software level requirements that are defined during top-level safety-related requirements considering the possible erroneous conditions such as the interface errors, logic errors, calculation or computation errors, or data errors [31].
- Perform risk assessment to reassess the risk in software level requirements and also determine the Safety Integrity Level (SIL) of the requirements in order to evaluate the criticality of each requirement.
- Propose mitigation approaches to manage the high risk and critical requirements based on the risk assessment results. The development teams need to decide whether treatment approaches should be proposed in the design.

2.1.4. Safety Specific Verification and Validation (V&V)

After the requirements are finalised, it is important for safety and requirements engineering teams to validate the correctness and completeness of the elicited safety-related requirements [10] according to the safety goals defined earlier. Any missing, incomplete, ambiguous, and incorrect requirements detected during this activity should be corrected. The corrections need to be analysed to ensure that the changes do not have negative impact on other requirements.

2.2. Phase II: Designing a survey to collect data from the experts

In Phase II, we had conducted a survey using a web-based questionnaire because it was more cost-efficient and the respondents can be geographically distributed across continents. The objective of the survey was to collect expert opinions in evaluating the validity of the selected SRATs and safety activities/steps. We had designed a structured questionnaire with close-ended, multiple-choice and open-ended type questions. The respondents can provide more detailed feedback in open-ended questions. The questionnaire consists of 15 questions.

It was a challenge to identify respondents for this survey because the respondents must have knowledge and expertise in both software safety and requirements engineering. We had sent our survey to 50 industry experts and researchers in various domains (e.g. aviation, oil and gas, nuclear, energy generation, railway, medical, communication and control) that have applied SRATs in eliciting software safety requirements. Out of the 50 requests that we sent, 29 experts answered the survey. The respondents had different experience levels and work functions in research or private sectors (e.g. safety engineers, system architects, technical advisors, process engineers, operation manager).

2.3. Phase III: Analysing data collected in Phase II

In Phase III, we analysed the survey data. The analysis results show that there are 93 percent of the respondents agreed that the requirements elicitation is an important activity in safety-related systems. However, there were only 55 percent of the respondents adopted a framework for requirements elicitation. Among these 55 percent of respondents, the existing industry standards are adopted as a framework and there are also guidelines to assist them in their elicitation activity.

All the techniques identified through literature (Table 1) were selected by the practitioners as suitable techniques for requirements elicitation (minimum three experts). Also, more than 50% of the experts had voted for the well-established techniques (i.e. PHA, safety checklists, FMEA/FMECA, and FTA). This supports the validity of the SRATs that we had selected from the literature are appropriate SRATs to be integrated into the elicitation activity.

We also analysed all the responses regarding the safety steps that have been carried out by the respondents' organisations in eliciting safety requirements. All the safety steps identified in Phase I are included. Although the respondents only voted for some of the steps, all of the steps have been considered as valid steps to be performed during requirements elicitation. Among all the steps, identifying safety goals, performing hazard analysis, performing risk assessment, and categorising requirements into specific levels are the common activities that have been carried out by most of the organisations. About 79% of the respondents agreed that integrating SRATs into requirements elicitation can help the requirements engineers, safety engineers and development teams to improve the elicitation of system and software requirements. This helps to develop safer software intensive systems.

3. Safety Risk Assessment Techniques In Requirements Elicitation (SaTRE) Integration Framework

3.1. Conceptual Metamodel

A conceptual metamodel of the SaTRE Integration Framework is developed based on Software Process Engineering Metamodel (SPEM) which comprises the basic process entities: *Activity*, *Work Product* and *Role*. This metamodel is extended by including two additional entities, *Technique* (i.e. SRAT) and *Step* (i.e. safety step). The conceptual metamodel is illustrated in Figure 1. As defined in the SPEM [49], an *Activity* may consist of one or more *Step*, whereas the *Work Product* represents the deliverables which are produced, used, or modified during the process. *Role* describes the responsibilities of stakeholders for each *Work Product* and also perform a particular *Activity*. *Technique* is a systematic procedure used to assist the *Activity*.

In this research, the *Activity* is the safety activity that has been integrated into appropriate requirements elicitation stages. Each safety activity consists of one or more detailed *Steps* defined to form the overall software safety process. *Work Product* is the artefact that is either the output of or the input from the safety activities and steps. The *Role* represents the main stakeholders who are responsible for the *Work Product* and at the same time perform the *Activity*. The *Technique* represents the SRATs applied to assist the stakeholders to perform specific *Activity* and *Step*. This metamodel forms the basis in designing the SaTRE Integration Framework. The relationships defined

between the entities in the metamodel are used to illustrate the activities in requirements elicitation.

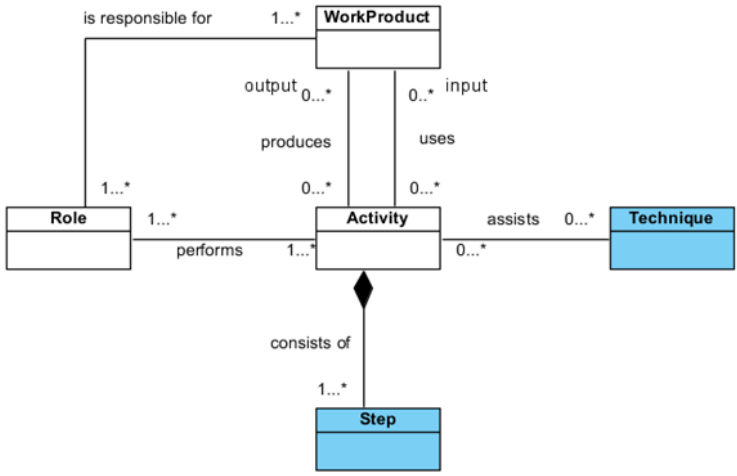

Figure 1. Conceptual metamodel of the SaTRE Integration Framework (adapted from SPEM [49]).

3.2. SaTRE Integration Framework

The overview of the SaTRE Integration Framework is shown in Figure 2. The requirements elicitation activity is divided into three stages: Pre-elicitation, Midst of elicitation, and Post-elicitation. The safety aspects are being considered in the requirements elicitation activity by integrating appropriate safety activities and SRATs into relevant elicitation stages. The aims of the pre-elicitation stage are identifying the goals and objectives of the project, the scope and system boundaries (e.g. domain) of the project and relevant requirements sources. After establishing required information from the pre-elicitation stage, the midst of the elicitation stage continues with the aims to elicit the existing requirements and the new requirements. The post-elicitation stage is carried out when all relevant stakeholders have agreed that there are no additional requirements or scenarios to be added.

In Section 2, a list of safety activities, steps and SRATs were selected. There are four main safety activities and each activity comprises a few different steps. These activities and steps are incorporated into Pre-elicitation, Midst of elicitation and Post-elicitation stages. Next, appropriate SRATs are integrated into these safety activities and steps. We have compiled a technique repository to store the information of the selected SRATs. This repository provides additional guidelines and templates to help the requirements and safety engineering teams to understand and apply the SRATs. The techniques repository also facilitates safety engineers to select techniques that are suitable for elicitation activity in their projects. Section 3.3 and Section 3.4 further describe the process of integrating SRATs into requirements elicitation activity based on this framework.

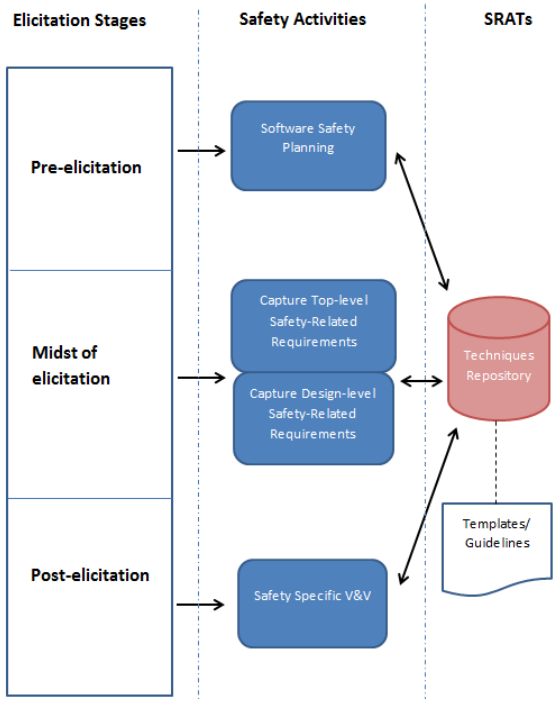

Figure 2. Overview of SaTRE Framework.

3.3. Assigning Safety Activities and Steps into Relevant Requirements Elicitation Stages

In this section, we mapped the safety activities/steps into appropriate elicitation stages by comparing the similarities of the objectives between requirements elicitation activity and the activities in software safety process. Also, we have assigned the safety activities to each respective stage according to the inputs and outputs of the safety activities.

The pre-elicitation stage takes place during the initial elicitation activity aims to define the project goals, determine the project scope/boundaries, and identify the relevant requirements sources of the system. We mapped software safety planning activity to this stage because both activities share the same objectives. Next, the midst of elicitation stage takes place whereby the existing and new requirements are elicited based on the identified requirements sources. We have mapped the safety activities that focus on identifying new safety-related requirements and proposing new mitigation approaches to this stage. The last stage of the requirements elicitation is the post-elicitation that validates and verifies the overall system requirements. From the safety aspect, safety specific V&V is performed to ensure that the defined safety-related requirements have fulfilled the safety goals. Both activities are mapped together to show the collaboration required by all the stakeholders in assuring the correctness and completeness of the requirements specification.

As shown in Figure 3, we integrate the safety activities into appropriate elicitation stages to ensure that safety quality is considered throughout the requirements elicitation process. Based on the expert opinions and literature survey, we have finalised the steps and assigned the software safety steps into the three stages of elicitation activity. This

helps the stakeholders (e.g. safety engineers, development teams) to understand the relationship between the RE and software safety processes.

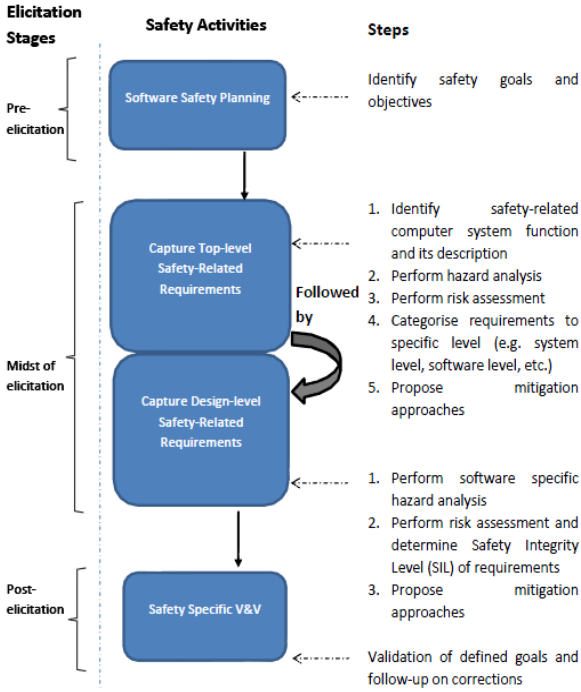

Figure 3. The Mapping Between Requirements Elicitation Stages and Safety Activities/Steps.

3.4. Integrating SRATs into Safety Activities and Steps

In this section, the SRATs are integrated into specific safety activities. We studied and reviewed the literature to identify the appropriate moment to apply each SRAT in the safety activities and steps during requirements elicitation to ensure that software safety risks are well identified and managed. Based on the characteristic of each SRAT and the objective of each safety activity, the SRATs are assigned to appropriate safety activities. Figure 4 shows the mapping between safety activities and techniques.

- Accident models [28, 29] identify the safety goals by analysing the physical factors and design practices that caused the accidents happened in the past. The development teams are able to discover flaws in the past practices and assess the risks of the system. Thus, this technique can be applied in software safety planning, capturing top-level and design-level safety requirements.
- PHA [22,24] is suggested to be applied during the identification of the safety goals because PHA can be used to define, analyse, and describe the hazards in the system at the initial stage. Information which is gathered initially such as the system boundaries, operational and environmental, and description about the system in the pre-elicitation stage.
- Safety checklists [23, 24] are to extract information from the past projects such as safety constraints that need to be considered or implemented in the new

project. Therefore, checklists can be used during software safety planning (i.e. the preparation of the safety plans), and also capturing top-level and design-level safety requirements (i.e. hazard analysis).

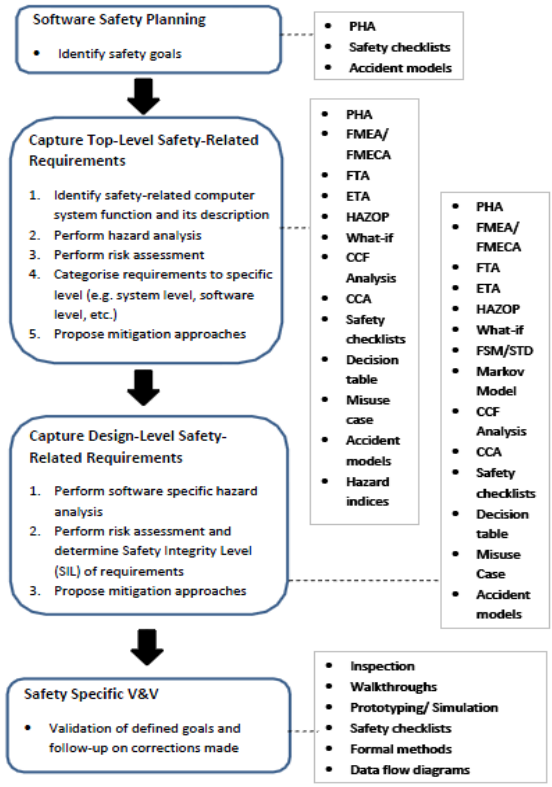

Figure 4. The Mapping Between Safety Activities and SRATs.

- What-if technique [21, 22] is a cost-effective brainstorming method used by stakeholders to analyse hazards and identify missing and incomplete top-level and design-level safety-related requirements in the midst of elicitation.
- To capture the top-level safety-related requirements, FMEA/FMECA [19, 24, 34, 35] and HAZOP [24, 36] are some of the SRATs that are commonly applied in identifying potential failure modes/deviation causes and its effects on other system components, and proposing mitigation actions for that particular failure mode.
- As for FTA [24, 34, 35, 37] and ETA [24, 34, 38], these techniques are used to identify the faults/events that would result in system failure and provide the probability of each failure. Therefore, both techniques can be applied for hazard analysis and risk assessment.
- The CCF analysis [24, 39] technique analyses the failure of several components and systems resulted from a cause or event within the specified system boundary. It is suitable to be applied during hazard analysis together with FTA that computes the probability of the failure.

- During hazard analysis, decision table [24, 40] can be used to analyse the situations where a combination of causes and the possible conditions would lead to different effects.
- CCA [19, 24, 41, 42] identifies the initiating events and conditions that lead to different consequences. Subsequently, risk assessment can be performed using the results from CCA.
- Hazard indices [29] provides information about well-known hazards. This technique can be used as a quantitative way of measuring the potential hazards. Hence, it can be used for hazards analysis and risk determination.
- For elicitating design-level safety-related requirements, FST/STD [24, 43] and Markov model [24, 46, 45] can be used when the initial designs are available. Both techniques can be used to predict the inputs and outputs of the transition between the system states. Software specific hazard analysis can be performed in early requirement phase to ensure the completeness of requirements.
- Inspection and walkthroughs [10, 24, 46] are two most widely used SRATs that allow stakeholders to verify and validate the correctness of the elicited requirements during the post-elicitation stage.
- Prototyping [24, 46] has been very useful in helping the development teams to validate the requirements by detecting the problems and clarify uncertainties in certain aspects of the system design.
- Formal methods [24, 47, 48] are mathematical modeling techniques that applied in safety-related systems to validate and verify whether the initial design of the systems have satisfied its safety properties.
- DFD [10, 24] can be used to illustrate the transformation of the input to output of data in the system after initial design-level requirements is defined.

A generic integration of SRATs, safety activities and steps into relevant requirements elicitation stages is presented in Appendix. Table 2 also shows the mapping of process entities such as work product and roles that are related to the SRATs. This helps the requirements or safety engineers to understand who can assists in performing the SRATs, and also know the deliverables which are produced, used, or modified during the process. A web-based tool (http://satre-tool.somee.com) was designed and developed according to the proposed framework to help requirements and safety engineers to apply this framework and customise the integration process based on individual project's needs.

4. Evaluation and Discussion

For the evaluation of the framework, an online questionnaire was created using Google docs and the link was distributed to the safety and RE experts via email. In the email, we also included a guideline of the SaTRE Integration Framework and a bibliography of the SRATs to help the RE and safety experts to understand and review the proposed framework. There are 9 experts reviewed the SaTRE Integration Framework and responded to the questionnaire. Five of them are from the safety-related industries (i.e. senior system engineers, operational manager, principal software architect, and process engineer) and the remaining four are researchers in the area of requirements engineering, dependability, and software safety. Based on the analysis results, the following are concluded:

- *Overall expert opinions* - Average of the experts have rated the framework in the range of moderate to good. Thus, this framework will be used as the basis for analysis and developing SaTRE web-based tool.
- *Validity of the mapping* - Experts review shows that the mapping of the requirements elicitation stages, SRATs, safety activities, and steps involved in the proposed framework are valid and acceptable.
- *Applicability*–The average response of the experts showed that the generic SaTRE Integration Framework can be used and customised for different projects. The identified process entities such as the input and output of the safety steps and activities, roles and techniques in the software safety process, and also the defined relationship between safety activity, techniques and requirements elicitation activity can be used to create method fragments to customise requirements engineering process. This framework comprises a set of SRATs that allows the requirements and safety engineering teams to decide which SRATs should be used to elicit software safety requirements in each stage. The proposed work can also be used to support the selection approaches which help the teams to choose SRATs systematically, e.g. based on a multi-criteria decision making method.
- *Limitations* – The main limitation is that the framework only caters the integration of SRATs into requirements elicitation. This research work can be extended to integrate the SRATs into other software development life cycle phases.

5. Conclusion and Future Work

The problem of incompleteness and the incorrectness of software requirements are some reasons that caused the software of safety-related systems to fail to achieve their safety goals. The two key factors i.e. software safety risks assessment and software safety requirements elicitation should be performed to ensure the overall safety of software intensive systems. Thus, in this research, SaTRE framework was proposed to help the safety and engineering teams to integrate appropriate software safety activities and techniques into requirements elicitation activity. The SRATs presented in this framework are not domain specific and they are applicable to most of the safety-related systems.

The framework was organised based on the three stages of requirements elicitation (pre-elicitation, midst of elicitation, and post-elicitation). Selected SRATs and activities were incorporated into the relevant stages of the requirements elicitation. This integration framework also can help the teams to choose the techniques that can be performed to elicit the specified level of software safety-related requirements. The integration is illustrated using a conceptual metamodel can be used to create method fragments for safety-related requirements elicitation activity. The expert's evaluation shows that the framework is useful to assist the new requirements and safety engineers in eliciting software safety requirements. A generic integration of SRATs, safety activities, and steps into requirements elictation was presented where requirements and safety engineers may refer if they need more detailed information on the application of the SRATs during requirements elicitation.

The current work focuses on the elicitation activity. This work could be extended to incorporate the safety process into other software development lifecycle phases. Subsequently, we will develop a tool to support both requirements and safety engineers in applying the proposed framework and providing some guidelines (such as the SRATs information, steps, roles and work products) that can help the stakeholders to understand the application of the SRATs.

References

[1] M. Ouyang, L. Hong, M.H. Yu, and Q. Fei, STAMP-based analysis on the railway accident and accident spreading: Taking the China–Jiaoji railway accident for example, *Safety Science* **48(5)** (2010), 544-555.
[2] T. Cant, Computer-based safety critical systems in defence: Def (Aust) 5679, *Proceedings of the 7th Australian workshop conference on Safety critical systems and software*, Australian Computer Society, Adelaide, Australia**, 152003** (2002), 9-16.
[3] J.L. Lions, *Ariane 5 flight 501 failure, Report by the enquiry board,* retrieved from http://www.cs.berkeley.edu/~demmel/ma221/ariane5rep.html, 1996.
[4] N.G. Leveson, *Medical devices: The therac-25*. Appendix of: Safeware: System Safety and Computers, 1995.
[5] R.R. Lutz, Software engineering for safety: a roadmap, *Proceedings of the Conference on The Future of Software Engineering*, ACM: Limerick, Ireland, 213-226, 2000.
[6] R.R. Lutz, Analyzing software requirements errors in safety-critical, embedded systems, *Proceedings of IEEE International Symposium on Requirements Engineering*, San Diego, CA, 213-226, 1993.
[7] E. Sikora, B. Tenbergen, K. Pohl, Industry needs and research directions in requirements engineering for embedded systems, *Requirements Engineering* **17(1)** (2012), 57-78.
[8] G. Kotonya, and I. Sommerville, *Requirements Engineering: Processes and Techniques*, Chichester, UK: John Wiley and Sons, 1998.
[9] B. Nuseibeh, and S. Easterbrook, Requirements engineering: a roadmap, in *Proceedings of the Conference on The Future of Software Engineering*, ACM: Limerick, Ireland, 35-46, 2000.
[10] K. Pohl, *Requirements Engineering: Fundamentals, Principles, and Techniques*, Springer Publishing Company, Incorporated, 35-36, 2010.
[11] D.G. Firesmith, Engineering Safety and Security Related Requirements for Software Intensive Systems, *Companion to the Proceeding of the 29th International Conference on Software Engineering*, Minneapolis, MN, USA, 169, 2007.
[12] G. Kotonya, and I. Sommerville, Integrating safety analysis and requirements engineering, *Asia Pacific and International Computer Science Conference*, Hong Kong, China, 259-271, 1997.
[13] *IEC61508-3: Functional safety or electrical/electronic/programmable electronic safety-related systems - Part 3: Software requirements*, 2010.
[14] *IEC62279: Railway Applications - Communications, Signalling and Processing Systems - Software for Railway Control and Protection Systems*, 2002.
[15] D. Alberico, J. Bozarth, M. Brown, J. Gill, S. Mattern, A. McKinlay VI, *Software safety handbook. A Technical and Managerial Team Approach*, International System Safety Society, 1999.
[16] *NASA software safety guidebook*, NASA Technical Standard, NASA-GB-1740.13, 2004.
[17] K. Allenby, and T. Kelly, Deriving safety requirements using scenarios, *Proceedings of the 5th IEEE International Symposium in Requirements Engineering*, 228-235, 2001.
[18] F. Giannini, M. Monti, S. Ansaldi, P. Bragatto, PLM to support hazard identification in chemical plant design Innovation in Life Cycle Engineering and Sustainable Development, in D. Brissaud, S. Tichkiewitch, and P. Zwolinski (eds.), *Innovation in life cycle engineering and sustainable development*, Springer Netherlands, 349-362, 2006.
[19] N.G. Leveson, *Safeware: system safety and computers*: Addison-Wesley. New York, USA, 1995.
[20] C.A. Ericson, *Hazard analysis techniques for system safety*: John Wiley and Sons. Hoboken, New Jersey, 2005.
[21] J. D. Lawrence, *Software safety hazard analysis*. Division of Reactor Controls and Human Factors, Office of Nuclear Reactor Regulation, US Nuclear Regulatory Commission, 1996.
[22] A.J. Card, J.R. Ward, P.J. Clarkson, Beyond FMEA: The structured what-if technique (SWIFT). *Journal of Healthcare Risk Management* **31(4)** (2012), 23-29.
[23] R.R. Lutz, Targeting safety-related errors during software requirements analysis, *SIGSOFT Software Engineering Notes* **18(5)** (1993), 99-106.

[24] IEC61508-7, *Functional safety or electrical/electronic/programmable electronic safety-related systems - Part 7: Overview of techniques and measures*, 2010.
[25] N.G. Leveson, Applying systems thinking to analyze and learn from events, *Safety Science* **49(1)** (2011), 55-64.
[26] N.G. Leveson, A new accident model for engineering safer systems, *Safety Science*, 42(4) (2004), 237-270.
[27] G. Sindre, A Look at Misuse Cases for Safety Concerns Situational Method Engineering: Fundamentals and Experiences, *Working Conference on Situational Method Engineering: Fundamentals and Experiences (ME07)*, 252-266, 2007.
[28] O.T. Arogundade, A.T. Akinwale, Z. Jin, X.G. Yang, Towards an Ontological Approach to Information System Security and Safety Requirement Modeling and Reuse, *Information Security Journal: A Global Perspective* **21(3)** (2012), 137-149.
[29] J., Suardin, M.M. Sam, El-Halwagi, Mahmoud, The integration of Dow's fire and explosion index into process design and optimization to achieve inherently safer design, *Journal of Loss Prevention in the Process Industries* **20(1)** (2007), 79-90.
[30] H. Azian, M. Yusof, A. Leman, Proposal of Development of Welding Health-Hazard Index (WHI) for Small and Medium Enterprises (SMEs), *ARPN Journal of Science and Technology* **2(2)** (2011), 62-67.
[31] D.P. Murray, and T.L. Hardy, Developing safety-critical software requirements for commercial reusable launch vehicles, *Proc. Of the 2nd IAASS Conference of Space Safety in a Global World*, Chicago, 2007.
[32] *IEC61508-0: Functional safety or electrical/electronic/programmable electronic safety-related systems - Part 0: Functional safety and AS 61508*, 2010.
[33] S. Kumari, G. Kondeti, S. Pakki, T.L.V. Chandrasekhar, and S. Balu, Method of safety critical requirements flow in product life cycle processes, *Integrated Communications, Navigation and Surveilance Conference (ICNS)*, N2-1-B2-4, 2011.
[34] S. Hosseini, and M. Takahashi, Combining Static/Dynamic Fault Trees and Event Trees Using Bayesian Networks, in F. Saglietti, N. Oster (eds), *Computer Safety, Reliability, and Security*, Springer Berlin / Heidelberg, 93-99, 2007.
[35] R.R. Lutz, R.M. Woodhouse, Contributions of SFMEA to requirements analysis, *Proceedings of the Second International Conference in Requirements Engineering*. Colorado Springs, 44-51, 1996.
[36] C. Liu, Y. Wang, W. Zhang, and Z. Jin, Elicitation of Dependability Requirements: A HAZOP-based Approach, *IEEE International Requirements Engineering Conference*, Sydney, NSW, 407-408, 2010.
[37] P. Katsakiori, G., Sakellaropoulos and E. Manatakis, Towards an evaluation of accident investigation methods in terms of their alignment with accident causation models *Safety Science* **47(7)** (2009), 1007-1015.
[38] O. Nývlt, and M. Rausand, Dependencies in event trees analyzed by Petri nets, *Reliability Engineering& System Safety* **104** (2012), 45-57.
[39] T. Lilleheier, *Analysis of commom cause failures in complex safety instrumented systems*. Norwegian University of Science and Technology, Norway, 2008.
[40] Y. M. Hon, *Decision tables and OBDDs*, Technical University of Braunschweig, Institute of Railway System Engineering and Traffic Safety, 2008.
[41] D.S. Nielsen, *The Cause/Consequence Diagram Method As A Basis For Quantitative Accident Analysis*. Danish Atomic Energy Commission, Risoe. Research Establishment, Denmark, 1971.
[42] *IEC/ISO-31010: Risk management – Risk assessment techniques*, 54-56, 2009.
[43] F. Wagner, Modeling software with finite state machines: a practical approach: Auerbach Publications, Boca Raton, New York, 2006.
[44] B. Knegtering, and A.C. Brombacher, Application of micro Markov models for quantitative safety assessment to determine safety integrity levels as defined by the IEC 61508 standard for functional safety, *Reliability Engineering & System Safety* **66(2)** (1999), 171-175.
[45] J.V. Bukowski, Using markov models to compute probability of failed dangerous when repair times are not exponentially distributed, *Annual Reliability and Maintainability Symposium*, Newport Beach, California, USA, 273-277, 2006.
[46] J. Thompson, M. Heimdahl, S. Miller, Specification-Based Prototyping for Embedded Systems, in *7th ACM SIGSOFT international symposium on Foundations of software engineering (ESEC/FSE-7)*, Springer-Verlag, London, UK, 163-179, 1999.
[47] Y. Junbeom, J. Eunkyoung, and C. Sungdeok, Formal Modeling and Verification of Safety-Critical Software, *IEEE Software* **26(3)** (2009), 42-49.
[48] J.P. Bowen, and M. Hinchey, Ten Commandments of Formal Methods... Ten Years On. *Conquering Complexity*, 237-251, 2012.
[49] OMG: Software & Systems Process Engineering Metamodel Specification (SPEM). Retrieved from http://www.omg.org/spec/SPEM/2.0/PDF, 2002.

Appendix

Table 2. A Generic Integration of SRATs into Requirements Elicitation Stages based on the SaTRE Integration Framework

Elicitation Stages	Safety Activities	SRATs	Basic Reference	Step	Work Product (Input, Output)	Role
Pre-elicitation	**Software Safety Planning** Identify safety goals	PHA	[22-24]	-	**Input** Artefacts (e.g. Safety standards, policies, procedures, manual, reports), Objective, Scope, Identified requirements sources, System boundaries **Output** Safety plan and goals	Safety analysts Requirements analysts Stakeholders
		Safety checklists	[23, 24]	-		
		Accident models	[28, 29]	-		
Midst of elicitation	**Capturing top-level requirements** **Steps:** 1. Identify safety-related computer system function and its description 2. Perform hazard analysis 3. Perform risk assessment 4. Categorise requirements to level (e.g. system level, software level) 5. Propose mitigation approaches	PHA	[22-24]	1,2	**Input** Artefacts, Safety plans, Safety goals, Use case scenarios, New/Existing requirements **Output** Top-level safety requirements, Probability of failure, Likelihood/ frequency of hazard occurrence, Risk assessment result, Mitigation approaches	Safety analysts Requirements analysts Safety analysts Safety experts
		FMEA/ FMECA	[34, 35]	2,3,4, 5		
		FTA	[24, 34]	2,3		
		ETA	[24, 34]	2,3		
		HAZOP	[24, 36]	2,5		
		What-if	[22]	1,2		
		CCF	[24, 39]	2		
		CCA	[15, 24]	2,3		
		Safety checklists	[23, 24]	1		
		Decision table	[24, 40]	2		
		Misuse case	[27]	2,5		
		Accident models	[28, 29]	1,2,3, 5		
		Hazard indices	[29]	2,3		
	Capturing design-level requirements **Steps:** 1. Perform software specific hazard analysis 2. Perform risk assessment and determine Safety Integrity Level (SIL) of requirements 3. Propose mitigation approaches	PHA	[22-24]	1	**Input** Safety plan, Safety goals, Use case scenarios, New/Existing requirements, Top-level safety-related requirements **Output** Design-level safety requirements, Risk assessment result, Requirements SIL, Mitigation approaches	Safety analysts Operators Safety experts Requirements analysts Domain experts Application designer / developer
		FMEA/FMECA	[34, 35]	1,2,3		
		FTA	[24, 34]	1,2		
		ETA	[24, 34]	1,2		
		HAZOP	[24, 36]	1,3		
		What-if	[22]	1		
		FSM/STD	[24, 43]	1		
		Markov Model	[44, 47]	1,2		
		CCF	[24, 39]	1		
		CCA	[15, 24]	1,2		
		Safety checklists	[23, 24]	1		
		Decision table	[24, 40]	1		
		Misuse Case	[27]	1,3		
		Accident models	[28, 29]	1,2,3		
Post-elicitation	**Safety Specific Verification and Validation** Assuring the software integrity and proper implementation	Inspection	[10, 24]	-	**Input** Safety goals, Safety plan, System/ Safety-related requirements, Use case scenarios **Output** Initial implementation plan, Test cases, Prototype	Safety analysts Requirements analysts External/ Internal validator
		Walkthroughs	[10, 24]	-		
		Prototyping/ Simulation	[24, 46]	-		
		Safety checklists	[23, 24]	-		
		Formal methods	[24, 47]	-		
		DFD	[10, 24]	-		